60100-0011 *Patent*

UNITED STATES PATENT APPLICATION

FOR

METHOD AND APPARATUS FOR ACCESSING AND MAINTAINING SOCKET CONTROL INFORMATION FOR HIGH SPEED NETWORK CONNECTIONS

INVENTORS:

JOHN MINAMI
ROBIN UYESHIRO
THIEN OOI
MURRAY WALLACE

PREPARED BY:

HICKMAN PALERMO TRUONG & BECKER LLP
1600 WILLOW STREET
SAN JOSE, CA 95125
(408) 414-1080

EXPRESS MAIL MAILING INFORMATION

"Express Mail" mailing label number EV323352304US

Date of Deposit April 12, 2004

METHOD AND APPARATUS FOR ACCESSING AND MAINTAINING SOCKET CONTROL INFORMATION FOR HIGH SPEED NETWORK CONNECTIONS

FIELD OF THE INVENTION

[0001] The present invention generally relates to accessing and maintaining socket control information for high speed network connections. The invention relates more specifically to a method and apparatus for maintaining socket information using a multiple port cache memory and controller.

BACKGROUND

[0002] The approaches described in this section could be pursued, but are not necessarily approaches that have been previously conceived or pursued. Therefore, unless otherwise indicated herein, the approaches described in this section are not prior art to the claims in this application and are not admitted to be prior art by inclusion in this section.

[0003] The packaging of data into data packets for network transfer and the reassembly of the data packets on the receiving end consumes a large amount of CPU cycles on a host computer system. As network bandwidth increases using higher speeds, such as the gigabit range, the workload on the host computer system increases. This causes the host computer system to spend more valuable CPU cycles managing network communications instead of executing application programs.

[0004] Transport control protocol (TCP) and Internet protocol (IP) are commonly used for packet network communications and are defined in Request for Comments (RFC) documents maintained by the Internet Engineering Task Force (IETF). TCP/IP or transport

offload engines (TOE) are used to move TCP/IP processing from the operating system to a specialized processor, often located on a network interface card (NIC). Adding a TOE can dramatically increase throughput. A TOE takes the job of translating all or part of the TCP/IP protocol away from the host's main processor, thus freeing up the host computer system to run other applications.

[0005] TCP/IP utilizes socket connections. The host computer system must notify the TOE of the sockets that it wants the TOE to handle. The host computer system transfers socket control information to the TOE. Some TOEs store the socket control information in a data structure in memory local to the TOE. During socket processing, the TOE references the socket control information to manage the socket communications. However, as the transfer speed across the network increases, the socket control information must be accessed faster and more efficiently.

[0006] Based on the foregoing, there is a clear need for a system that provides for the management of socket control information for high-speed networks in a highly accessible manner. Additionally, there is a need for a system that would allow operations such as transmit and receive to reference socket control information at a higher priority than other operations.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] Embodiments are illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

[0008] FIG. 1 is a block diagram that illustrates a CPU in communication with a transport offload engine (TOE) interfacing with an Ethernet network according to an embodiment of the invention;

[0009] FIG. 2 is a block diagram that illustrates a TOE structure according to an embodiment of the invention;

[0010] FIG. 3 is a block diagram that illustrates a multi-port cache structure with a cache controller arbitrating between multiple access interfaces according to an embodiment of the invention;

[0011] FIG. 4A is a flowchart that illustrates locating a CB entry in the CB cache using a hash reference table when no external CB memory is present according to an embodiment of the invention;

[0012] FIG. 4B is a block diagram that illustrates the use of a hash reference table and a CB look up table (LUT) according to an embodiment of the invention;

[0013] FIG. 5 is a flowchart that illustrates locating a CB entry in the CB cache using a hash reference table when external CB memory is present according to an embodiment of the invention;

[0014] FIG. 6 is a flowchart that illustrates locating a CB entry in the CB cache using a CB handle reference table when no external memory is present according to an embodiment of the invention; and

[0015] FIG. 7 is a flowchart that illustrates locating a CB entry in the CB cache using a CB handle reference table when external memory is present according to an embodiment of the invention.

DETAILED DESCRIPTION

[0016] A method and apparatus for accessing and maintaining socket control information for high-speed network connections is described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

[0017] Embodiments are described herein according to the following outline:

* * *

1.0 GENERAL OVERVIEW

[0018] The needs identified in the foregoing Background, and other needs and objects that will become apparent for the following description, are achieved in the present invention, which comprises, in one aspect, a method for accessing and maintaining socket control information for high speed network connections.

[0019] A multi-port control block ("CB") cache contains socket control information in CB entries for sockets assigned to the TOE by a host computer. The socket control information is needed by the TOE to form transmit packets and check if receive packets are assigned to the TOE, for example.

[0020] Each port provides a TOE client direct access to the CB cache. Time critical TCP receiving (RX) and TCP transmitting (TX) logic are each provided dedicated ports to enable higher bandwidth accesses to the CB cache. All other non-time critical TOE clients are given arbitrated access via a separate dedicated port.

[0021] An optional external memory is given direct access to the CB cache via a dedicated port. The external memory can store more socket control information in CB entries for additional sockets. When the optional external memory is present, the CB cache keeps one CB entry empty to facilitate fast CB entry transfers between the external memory and the CB cache. The CB cache transfers a CB entry from the external memory before writing an older CB entry to the external memory, thus freeing up the older CB entry's slot in the CB cache.

[0022] When the optional external memory is not present, the same dedicated port mentioned above may be used by the host to directly transfer CB information to and from the CPU and the TOE.

[0023] Arbitration between clients that share a port on the CB Cache is based on a priority model where some clients are assigned a higher priority than other TOE clients. The priority scheme is based on how time-critical of each clients accesses to CB information is.

[0024] CB entries are located in the CB cache using two data structures: a hash reference table and a CB identifier (also referred to herein as a CB handle) reference table. The hash reference table is used by the RX logic when it is trying to find a CB entry associated with a

received packet. The CB identifier reference table is used by all other TOE clients when the CB handle is already known.

[0025] An access locking mechanism is provided to ensure CB entry coherency. Access locks are requested per CB entry field, which allows multiple TOE clients to access the same CB entry simultaneously, but not just the same field within the CB entry. CB entry accesses may be of the locked or unlocked variety.

[0026] In addition, when external memory is present for additional CB entry storage, the CB Cache controller uses a least recently used (LRU) algorithm to determine which CB slot to write back to external memory when a new CB entry is read into the cache.

2.0 STRUCTURAL AND FUNCTIONAL DESCRIPTION

2.1 TRANSPORT OFFLOAD ENGINE (TOE)

[0027] Fig. 1 illustrates a host computer CPU 101 in communication with a TOE 102. The TOE 102 is communicably linked to a local area network (LAN) 103 such as an Ethernet network, or any other data link layer protocol. The TOE 102 can be located on the same circuit board or silicon as the host CPU 101, or on a network interface controller (NIC) board or card. The CPU 101 has its network communications offloaded by the TOE 102. The CPU 101 passes the TOE 102 the communication socket control information for the sockets that the CPU 101 wants the TOE 102 to handle.

[0028] The TOE 102 manages the sockets that the CPU 101 has assigned to the TOE 102 as network packets move between the network 103 and the CPU 101. The TOE 102 uses the socket control information to manage transmit and receive packets over the network 103. To be able to keep up with the high data transfer rates of the network 103, the TOE 102 manages

the socket control information located in the TOE 102 using a multiple-port control block (CB) cache memory and controller. The multi-port CB cache memory and controller allow the TOE 102 to service transmit and receive packets at a rate that is more than adequate to keep up with transfer speeds of 10Gb and above.

[0029] Fig. 2 illustrates a high-level view of an embodiment of the invention incorporated into a TOE 201. The TOE 201 shown supports dual media access controllers (MACs) 208, 209. However, any number of MACs can be provided in alternative embodiments, and two is not limiting. Transmit and receive buffers can be incorporated between the MACs 208, 209 and an integrated network interface 207 to allow any type of MAC to be incorporated into the TOE 207.

[0030] A host interface 202 allows the TOE 201 to transfer data to and from a host CPU.

[0031] The main sections of the TOE 201 are the transmit (TX) 203 and receive (RX) 204 modules. The TX 203 module processes data packets sent by the host to the TOE over the host interface 202. The RX module 204 processes incoming data packets from the Ethernet interface 207.

[0032] The control block (CB) cache 206 contains socket control information for sockets that have been offloaded from the host CPU. The CB cache controller 205 allows multi-port access to the CB cache 206. Multiple ports enable an embodiment of the invention to provide access to the CB cache 206 at speeds sufficient to handle 10Gb data rates and above.

[0033] A port on the CB cache is dedicated for communications with an optional external memory 210. This memory is used to store addition CB entries. When the external memory is not present, the port on the CB cache is used by the host to transfer CB entries to and from the host system and the TOE.

2.2 CONTROL BLOCK (CB) CACHE

[0034] An embodiment of the invention uses a cache structure (CB cache 206) that works in conjunction with an optional external memory structure. The CB cache 206 is equipped with multiple ports to allow greater access bandwidth to resources that are time critical for processing network traffic. The CB cache 206 contains socket control information and, alternatively, can be used to store other data structures *e.g.*, instruction block queues. This allows the entire TOE 201 to keep up with very high-speed network traffic (10Gb and above).

[0035] Fig 3. illustrates an embodiment of the invention using a multi-port SRAM for the CB cache 301. Each CB entry in the CB cache 301 contains socket control information such as: the state of the socket, remote IP address and port, remote MAC address, operating status of the socket (sliding window, delayed ACK mode, etc.), sequence numbers, ACK numbers, etc. The socket control information is needed by the TOE to form transmit packets and check if receive packets are assigned to the TOE, for example. The socket control information in the CB cache 301 is updated by components (TOE clients) in the TOE that track the sockets and their status. TOE clients can be any component that operates within the TOE or any component that can externally access the TOE. The CB cache controller 302 is responsible for managing and arbitrating accesses to the CB cache 301.

[0036] The CB cache 301 is needed because the optional external SRAM 306 where the main CB memory is typically located is shared with other TOE data structures (SGLs, FIFOs, Queues, etc.). Therefore, in order to meet the access requirements needed to support 10Gb network speeds, a fast access method is implemented that allows the different CB clients enough bandwidth to service both RX and TX packets. When the external SRAM 306 is not present, the CB cache 301 serves as the main CB memory.

[0037] In one embodiment, the CB cache 301 is implemented as a quad port, 512 x 64bit, synchronous SRAM. This is enough memory to hold 16 CB entries with each entry being 256 bytes in length. In other embodiments, the CB cache 301 may have any other size appropriate for the application. The CB cache controller 302 accepts CB access requests from various resources, checks to see if the desired CB is in the cache, and if it is, arbitrates between all good requests for accesses to the CB cache 301. Time critical TCP RX and TCP TX accesses go through their own interfaces 305, 304 to enable higher bandwidth accesses to the CB cache 301. All other non-time critical TOE clients are arbitrated through a third interface 303.

[0038] Each interface has a dedicated port to speed accesses to the CB cache 301. The RX access interface 305 is communicably connected to port A 307, the TX access interface 304 is communicably connected to port B 308, and the arbitrator for other accesses 303 is communicably connected to port C 309. A fourth port, port D 310, is provided to allow the CB cache controller 302 to transfer data to and from the host (also a TOE client) or an optional external SRAM 306. This allows the CB cache controller to retrieve new CB entries or store old CB entries while simultaneously granting access to TOE clients to other CB entries within the CB Cache.

[0039] Access arbitration within each port is based on a predetermined priority model where some clients (*e.g.*, the RX packet parser and TX packet generator modules) are assigned a higher priority than other TOE clients. The priority assigned to each client is determined by how time critical the TOE client's CB accesses are.

[0040] An external SRAM 306 may be added to expand the capabilities of the TOE. The external SRAM 306 can store more socket control information for additional sockets. An embodiment of the invention can detect the presence of the external SRAM 306 via a

configuration resistor, detecting the external SRAM address lines, or other methods. If the external SRAM 306 is not present, then the CB cache 301 functions as the main CB memory.

2.21 CB CACHE BANDWIDTH

[0041] The CB cache structure is designed to support dual 10Gb Ethernet links. Therefore, the total bandwidth required for full duplex line speed is 40Gb/second (10Gb/sec x 2 (Links) x 2 (for TX and RX), or 5GB/second. Assuming that an average packet requires 128 bytes , the total packet throughput is:

[0042] Packet Throughput = 5GB/128 = 39.1 M Packets/second

[0043] Further, assuming that 128 bytes are accessed per packet, the total CB access requirement is:

[0044] CB Access Requirement = (39.1 M Packets/second) x (128 Bytes/Packet) = 5 GB/second

[0045] The requirement covers both TX and RX directions. Since they are given dedicated ports 307, 308 into the CB cache 301, each port in the cache requires a 2.5GB/second bandwidth. The target operating frequency is 200 MHz and the bus width for the CB cache (to internal TOE clients) is 128 bits. Therefore, the theoretical bandwidth available to each of the TX and RX clients is:

[0046] Theoretical Bandwidth = (200 M x 128 bits) / (8 bits/Byte) = 3.2GB/second

[0047] Therefore, the dedicated ports for TX 308 and RX 307 will provide enough bandwidth access to the CB cache to support dual 10Gbps Ethernet links. In fact, the dedicated ports will support the links at line speed.

[0048] All other CB clients that are non-time critical will share another port 309 into the CB cache 301.

2.22 CB ENTRY LOOKUP USING SOCKET HASHES

[0049] Figs. 4A and 4B illustrate an embodiment where CB entries are located in the CB cache using a hash reference table when no external memory is present. The hash reference table is needed because when a network packet is received, the RX logic does not yet know the CB handle for the socket, or whether the packet is associated with a socket assigned to the TOE. If a CB entry is found that contains the same socket parameters (IP addresses and port numbers) as the received packet, then the packet is processed by the TOE. If no matching CB entry is found, then the packet is forwarded to the host for processing.

[0050] The TOE's RX logic first generates a hash value for each received network packet as indicated in operation 402. The hash is based on the four-tuple parameters of the packet (the local and remote IP addresses, and the local and remote ports). The received network packet hash value is then compared to the hash values in the hash reference table (452) in operation 404. The CB Cache controller will then return a bitmap indicating which slots (if any) in the CB Cache (456) have matching hash values. See operation 406. A check is then made in decision 408 to see if any of the CB entries in the CB cache have hashes that match that of the received network packet. If there are no matches and since external memory is not present, then the TOE has not been assigned the received network packet's socket. See operation 422. The packet is then forwarded to the host for processing in operation 424. If there are matches, then the RX logic will then read the IP addresses and port numbers from the first matching CB entry in the cache to see if they match those of the received network packet. See operation 410 and 412. If they do match, then the correct CB entry has been found. See decision 414 and operation 416. If the parameters do not match, then a check is made to see if there are more matching CB entries in the CB cache. See decision 418. If

there are more entries, then the next set of socket parameters is read in operation 420, and compared against those of the received network packet in operation 412. If there are no more matching entries, and the correct CB entry has not yet been found, then the packet does not belong to a socket that has been assigned to the TOE (operation 422), and the packet is again forwarded to the CPU for processing. See operation 424.

[0051] Fig. 5 depicts an embodiment that locates CB's when external memory is present. The RX logic begins by again generating a hash for the received network packet in operation 502. It will then simultaneously query the CB Cache controller's Hash reference table (452) (operation 522) as well as a CB look up table (LUT) (454) (operation 504). The CB LUT (454) uses the hash generated from the received network packet as an address into the LUT and returns a valid indicator and a CB handle (460). If the CB LUT returns a valid indicator (see decision 506), then the socket 4-tuple is read from the CB entry indicated by the CB LUT in operation 510. If the parameters match (see decision 512), then the CB has been found (see operation 514). Hash values are not guaranteed to be unique across all sockets assigned to the TOE. To handle the case of hash collisions (i.e. two or more sockets have the same generated hash value), a field is provided in the CB entry that points to the next CB handle that has the same hash value. This is referred to as the Linked CB handle. If the parameters did not match in decision 512, then the linked CB field is read from the CB entry in operation 516. If the CB field is not valid (see decision 518), it again means that the packet is associated with a socket not assigned to the TOE (see operations 532 and 534). If the linked CB field is valid, then the socket 4-tuple is read from the linked CB entry (462). See operation 520. The parameters are then again checked to see if they match the parameters from the received network packet. See decision 512. This continues until the CB is found or a CB's linked field is not valid.

[0052] This CB look up in external memory and the CB cache via the CB LUT is done in parallel to querying the CB Cache via the HASH reference table. If the query to the CB cache(see operation 524) completes and the CB entry is found to be in the cache (see decision 526), then the search for the CB via the CB LUT is terminated (see operation 528). It is therefore obvious that the CB Cache and CB location method outlined above provides a much quicker method for determining if a received packet is associated with a socket assigned to the TOE. If the CB entry is not found, then the process waits for the results of the CB LUT search (see operation 530).

[0053] If the CB LUT query in operation 504 does not return a valid LUT entry, than it also means that the packet is associated with a CB not assigned to the TOE (see operation 532). In this case, the packet is also forwarded to the host for further processing (see operation 534).

2.23 CB ENTRY LOOKUP USING CB HANDLES

[0054] Fig. 6 illustrates one embodiment where CB entries are located in the CB cache using a CB handle reference table (464) when no external memory is present. This method of CB access is used by clients that already know the CB handle they wish to access. An example of this type of client is the TX data packet generator. The module will get a request from the host to transmit data on a given socket, and the CB handle associated with the socket is passed to the TX data packet generator as part of the transmit request.

[0055] For these accesses, the TOE client begins by providing a CB address to the CB Cache controller (see operation 602). The address is comprised of the CB handle concatenated with a CB offset. For example, if each CB entry is made up of 64 double word fields, then the CB offset is 6 bits long. If the CB handle is 16 bits long (thus allowing for

64K CB's), then if a client wanted to access the second DWORD in a CB entry with handle 0x1005, the address calculation is:

[0056] CB Address = {CH_Handle, CB_Offset} = {16'h1005, 6'h02} = 22'h040142

[0057] The TOE client begins the request by providing the CB address to the CB Cache controller in operation 602. The CB Cache controller then parses out the CB handle from the supplied address in operation 604. Using the above example, the CB handle is obtained by parsing bits[21:6] of the CB address. The CB Cache controller then queries the CB Handle reference table (464) to find which CB cache slot (456) contains the CB handle requested. See operation 606. If no matching CB handle is found in the reference table (per decision 608), then an error condition (612) is reported back to the host as it has referenced a socket that is not assigned to the TOE. Normally, the CB Handle reference table will return a CB slot where the CB entry is located, at which point the CB Cache controller can grant the TOE clients access request to the CB. See operation 610.

[0058] Fig. 7 illustrates an embodiment where CB entries are located in the CB cache when external memory is present in the TOE system. In this case, the TOE client again begins by providing a CB address to the CB Cache controller. See operation 702. The CB Cache controller parses out the address in operation 704 in the same manner as outlined above. The CB Handle is then compared to entries in the CB Handle reference table (464). See operation 706. Then, depending on whether the CB handle matched any entries in the reference table (see decision 708), the TOE Client is either granted access to the CB (see operation 714) or the CB is fetched from external memory (458). See operation 710. After the CB is fetched, the TOE client is given access to the CB in operation 714, and in parallel, the least recently used (LRU) CB entry is written back to external memory. See operation 712.

[0059] As noted above, and in Fig. 3, in one embodiment, the CB cache 301 has space for 16 complete CBs. In this embodiment, all 16 spaces are used for CB entries when the external SRAM is not present. Any CB entries not in the CB cache 301 are sent to the host for the host to handle.

[0060] When the external SRAM 306 is present, a method is implemented that allows CB entries from the external SRAM 306 to be written to the CB cache 301 as soon as possible. In this method, of the 16 spaces in the CB cache 301, 15 are utilized at any given time. If the CB entry is in the CB cache 301, then the TOE client request can be serviced directly. The last or 16^{th} slot is saved for the case where a new CB entry needs to be fetched from the external SRAM 306. The new CB entry can be fetched first (see operation 710) with the TOE client being serviced immediately (see operation 714). Then, using a separate background process or thread, the least recently used CB entry is written from the CB cache 301 to the external SRAM 306 (see operation 712).

[0061] In addition, by knowing a client's access pattern (which is predetermined by the client), the WORD accessing order by the CB cache 301 is tailored such that the client can start reading the CB entry from the CB cache 301 while other parts of the CB entry are still being fetched from the external SRAM 306. After the CB entry has been completely fetched, the least recently used CB entry is written from the CB cache 301 to the external SRAM 306.

[0062] In parallel to detecting if the requested CB entry is in the CB cache 301, another look up is performed in order to determine if the field in the CB entry being requested is currently free for access or whether it is locked by another resource. A resource may need to lock a CB field if it is doing a read-modify-write operation. To detect this condition, the CB cache address being requested is sent to a sub-module which keeps track of which fields are locked. In the sub-module, the address is compared to see if any other resource is currently

locking the field. If no resources are locking the particular CB field, then the access cycle is allowed to proceed.

[0063] This CB field locking feature supports providing the maximum bandwidth to the CB cache 301 while still maintaining CB content coherency. If entire CB entries were locked out until a client was finished using them, then too many clients would spend too much time waiting for CB entry accesses. Likewise, if a locking mechanism is not provided, then it is possible for multiple clients to access the same field within a CB entry and CB content integrity would be compromised.

2.24 CB CACHE ORGANIZATION

[0064] The CB cache uses a least recently used algorithm for determining which CB is written back to external SRAM when a new CB needs to be fetched from external SRAM. To implement this, the CB cache controller maintains a list of the CB's and orders them according to accesses.

[0065] In an embodiment with 16 entries in the CB cache, after any reset, the list is established in the following order:

0	CB Slot 0 (oldest)
1	CB Slot 1
2	CB Slot 2
3	CB Slot 3
4	CB Slot 4
5	CB Slot5
6	CB Slot6
7	CB Slot7
8	CB Slot8
9	CB Slot9
10	CB Slot10
11	CB Slot11
12	CB Slot12

13	CB Slot13
14	CB Slot14
15	CB Slot15

[0066] When the first CB entry is accessed, it will go into slot 0. At that time, CB Slot 0 is moved to the bottom of the list, and all other entries will move up one slot. The list will appear as follows:

0	CB Slot1
1	CB Slot2
2	CB Slot3
3	CB Slot4
4	CB Slot5
5	CB Slot6
6	CB Slot7
7	CB Slot8
8	CB Slot9
9	CB Slot10
10	CB Slot11
11	CB Slot12
12	CB Slot13
13	CB Slot14
14	CB Slot15
15	CB Slot0

[0067] After 16 or more CB's are allocated, then the cache is full. At that time the list may have a random order depending on which CB's were accessed in what order. One entry is always kept open however. This is to allow a new entry to be read first thereby allowing quicker access to the new CB. After the new CB is read, then the LRU CB is written back to external memory. A sample list is shown below:

0	CB Slot2 (Empty)
1	CB Slot3 (oldest)
2	CB Slot6
3	CB Slot8
4	CB Slot10
5	CB Slot1
6	CB Slot15

7	**CB Slot4**
8	CB Slot5
9	CB Slot12
10	CB Slot11
11	CB Slot13
12	CB Slot0
13	CB Slot14
14	CB Slot7
15	CB Slot9

[0068] If at this time, CB Entry 4 (currently in the 7th position in the above list) is accessed, then it will move to the bottom of the list, and the list will now appear as shown below:

0	CB Slot2 (Empty)
1	CB Slot3
2	CB Slot6
3	CB Slot8
4	CB Slot10
5	CB Slot1
6	CB Slot15
7	CB Slot5
8	CB Slot12
9	CB Slot11
10	CB Slot13
11	CB Slot0
12	CB Slot14
13	CB Slot7
14	CB Slot9
15	**CB Slot4**

[0069] Next, if CB entry 8 is deprecated, it is moved up to the top of the list, and the list will now appear as shown below:

0	**CB Slot 8 (Empty)**
1	CB Slot 2 (Empty)
2	CB Slot 3
3	CB Slot 6
4	CB Slot 10
5	CB Slot 1
6	CB Slot 15
7	CB Slot 5
8	CB Slot 12
9	CB Slot 11
10	CB Slot 13
11	CB Slot 0
12	CB Slot 14
13	CB Slot 7
14	CB Slot 9
15	CB Slot 4

[0070] If a CB update request is made, then the oldest CB entry (in this case Entry 8 from the above table) is moved down to the bottom of the list. The list would then appear as follows:

0	CB Slot 2 (Empty)
1	CB Slot 3
2	CB Slot 6
3	CB Slot 10
4	CB Slot 1
5	CB Slot 15
6	CB Slot 5
7	CB Slot 12
8	CB Slot 11
9	CB Slot 13
10	CB Slot 0
11	CB Slot 14
12	CB Slot 7
13	CB Slot 9
14	CB Slot 4
15	**CB Slot 8**

3.0 EXTENSIONS AND ALTERNATIVES

[0071] In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.
